1-1-1976

The Earth Resources Interactive Processing System (ERIPS) Image Data Access Method (IDAM)

A. E. Pape

D. L. Truitt

Reprinted from

Symposium on

Machine Processing of

Remotely Sensed Data

June 29 - July 1, 1976

The Laboratory for Applications of
Remote Sensing

Purdue University
West Lafayette
Indiana

IEEE Catalog No.
76CH1103-1 MPRSD

THE EARTH RESOURCES INTERACTIVE PROCESSING SYSTEM (ERIPS)

IMAGE DATA ACCESS METHOD (IDAM)

A. E. Pape and D. L. Truitt

International Business Machines Corporation
Federal Systems Division
Houston, Texas

I. ABSTRACT

The Image Data Access Method (IDAM) was developed for NASA at the Johnson Space Center as the access method for the Earth Resources Interactive Processing System (ERIPS). It was designed to satisfy the need for an image data storage technique which is very efficient with storage space and data access time, but which also provides direct retrieval of all of the image data and easy programmer utilization. A set of eight user macros have been provided which satisfy the utilization requirement, and the result has proven to be a very successful image data storage and retrieval technique.

II. INTRODUCTION

Remotely sensed data in general, and that collected by multispectral scanners in particular, consist of large volumes of multi-dimensional data. The processing and analysis of this data normally depends on its consisting of groups or fields of data which are alike; i.e., intensity readings from a single object type. However, there are often more than one group of each object type within each image, and the data from all fields of each type need to be processed as if they were a single group when signature definition; i.e., class statistics calculations, is to be performed. The magnitude of the data and the field structure nature of the data require special storage and retrieval techniques in order to allow data analysis and information extraction using EDP equipment, without data magnitude and nature being a limiting factor in the processing. The storage and retrieval technique must contend with large volumes of data, which prohibits main core storage or even direct access storage of all of the data until it is needed for processing. But when processing begins, the storage and retrieval system must provide quick and easy access to data from like fields which may be in widely differing segments of a large image, so that they may be processed almost simultaneously.

Largely as a method of enhancing processing speed by reducing the number of calculations, the state-of-the-art in Earth Resources data analysis and information extraction also requires the capability to select or eliminate bands or channels of multispectral data in response to machine-aided or external feature selection determinations. In addition to the computational savings from channel elimination, the results for a particular analysis technique are often improved if only a subset of the available data is used in the analysis. In some cases, a particular analysis technique could not be used at all if channel elimination were not possible. For example, a computation involving the inversion of a covariance matrix would not be possible on data having correlated channels.

All of these considerations place demanding requirements on any access method which opts to serve as an image data storage and retrieval system.

III. CONVENTIONAL ACCESS METHODS

An analysis of the data access methods available with general purpose EDP hardware reveals that none of them can provide the wide variety of accessing options required by typical data analysis programs.

Sequential access methods provide only the capability to read entire data files in order, and offer no assistance to the program which wishes to process only selected records or parts of records from the file. Thus, the program must read to the desired starting point within the file, read the records from that point, and select the desired records or parts of records to be processed. This presents a considerable I/O overhead to the analysis program, particularly if the data is to be processed in a natural order (i.e., by channels and fields), because multiple reads of the same data are required.

Indexed sequential access methods provide the capability to start at a key-selected record and retrieve all records in key-order from that point. This method avoids the necessity of reading to the start of the desired data, but provides no assistance in selecting records from that point. Also, the reading and writing of keyed files is, in general, slower than reading and writing unkeyed files.

Direct access methods provide the capability to retrieve any one record by relative position within the data set. The direct method thus allows complete record selection freedom. But since the accesses can be random, no look-ahead reading by the

access method is possible. A further problem is that the analysis program must spend considerable effort in keeping track of the next record to be read.

None of the standard access methods with which we are familiar, other than very complex and consequently very slow data base management systems, provide any facility to retrieve only a part of a record or only selected records automatically. These two facilities plus the maximum attainable access rate were considered by the designers of the ERIPS to be essential to the development of data analysis programs with minimum complexity and repetition of code which would execute at or near the CPU bound rate.

IV. IDAM PHILOSOPHY

Since the required facilities could not be obtained in any existing readily available access method, NASA and IBM made the decision to develop a specialized Image Data Access Method (IDAM) for ERIPS. The ERIPS data handling techniques utilize both tape and direct access storage of image data. The bulk storage medium is computer-compatible tapes (CCT's). When an image is required for processing, it is read from the CCT using conventional access methods and written to disk via IDAM. All data analysis programs then access the data from disk by use of IDAM. Besides moving the required data from a low speed to a high speed medium, this process also transforms the data from the various CCT formats (LARSYS II, S-192, ERTS-BULK, ERTS-PRECISION, UNIVERSAL) into the single IDAM format.

The IDAM storage sequence is by line and by channel within lines. All pixels of channel-one, line-one, referred to later as a channel-line of data, are stored first, followed by all pixels for channel-two line-one. All channels of line-one are followed by all channels of line-two and so on.

The data is stored on the device in the largest physical blocks the device will support, and logical channel-lines of data are divided into more than one physical block as needed.

As IDAM writes an image to the device, control information is written with the data which allows the physical block containing any specified channel-line to be easily identified. This information includes (1) a channel mask specifying which channels are present in the images, (2) the number of pixels in each line, (3) the number of lines in the image, (4) the number of bits in each pixel, (5) line and pixel skip factors (explained below), (6) the line and pixel numbers of the first pixel image, and (7) the number of bytes of file and line ancillary data stored with the image.

The basic retrieval technique is straightforward. Using the information described above, IDAM computes the relative track number and displacement within that track of the requested data. IDAM then builds a set of I/O commands (a channel program) to (1) position the disk's read-head to the proper track, (2) skip over the part of the track which precedes the desired data, and (3) read the desired data into a main storage buffer from which it is moved into the requesting program's storage area.

By use of this basic technique, an access method has been constructed which allows a user program to quickly and easily retrieve any desired subset of an image, in terms of only selected channels, lines, and pixels within lines.

V. IDAM TECHNIQUE

Once the disk read-head is in position, all of the data, or any subset of the data, from a single track can be transferred into main storage buffers in one revolution of the disk, provided the data is being read sequentially by a single channel program. Typical disk track sizes range from approximately 7000 bytes to approximately 13000 bytes per track. There are few images for which several channel-lines of data can not be contained within one track. In order to optimize the speed of the access method whenever possible all of the individual channel program sequences, which read data from the same track by positioning the read-head to the start of the track, skipping to the desired channel-line of data and then reading the desired data, are combined into a single sequence which: (1) positions the read-head to the track, (2) skips to the first data segment to be read (of course, if the data start coincides with the beginning of the track, this step is skipped), (3) reads the desired data, (4) skips to the next data segment (provided that there is data between the desired data segments which much be skipped; otherwise, this step is skipped), (5) read the next segment of data, (6) repeat steps (4) and (5) until the end of the track is encountered. To further optimize the channel program when it is found that no data is to be skipped between consecutive reads; i.e., when step (4) is not needed, the consecutive data reads are combined into a single read of all of the data.

Although the user has the option of specifying the size and number of data buffers used by IDAM, the normal procedure is to use two buffers of default size; i.e., 2K or the smallest buffer that will hold one channel-line of data, whichever is largest. Sequences of channel program instructions, as described in the previous paragraph, are combined into a single channel program to fill one buffer. The channel program is then complete and ready to be executed to actually retrieve the data. As the channel program to be used in filling a buffer is being prepared, the appropriate buffer pointers and channel-line lengths are calculated and stored in the buffer's header which allows direct addressing of each channel-line of data in the buffer so that it can be quickly transferred to the users storage area when it is requested.

VI. IDAM UTILIZATION

The IDAM philosophy and technique are straightforward; however, they can seem rather confusing to the programmer who is inexperienced in writing

channel programs. Consequently, a set of eight user macros were prepared. The macro parameters are usually readily available apriori image or field characteristics. Following is a list of the macros, the functions they perform, and a typically coded macro call.

ICLOSE - Issued when the creation or utilization of an image data set on disk is complete. ICLOSE flushes all of the output buffers and issues an OS-CLOSE for the data set.

```
ICLOSE ISCBNAME
```

IDIRECT - Allows direct accessing and/or updating of any information in an existing image data set on disk, independent of a NEXTL and image buffering, provided that no storage-size increase is required by the updates.

```
IDIRECT ISCBNAME,WORKAREA,GET,MFANC,USERAREA,
        LENGTH
```

IGET - Allows the user to retrieve into his storage area one channel's data for one line; i.e., one channel-line, of the image (or field) as specified in the NEXTL. Subsequent IGET's retrieve the next channel-line of data as specified in the NEXTL.

```
IGET ISCBNAME,USERAREA
```

IOPEN - Allocates space for a new image data set or locates an old data set on disk. An OS-OPEN is issued for the data set, and buffers and work areas are obtained for processing the image. If the data set is new, IOPEN creates a pseudo-label for the data set which contains the ISCB information which will be needed the next time the image is IOPEN'd.

```
IOPEN ISCBNAME,INPUT,IMAGNAME
```

IPUT - Allows the user to output from his storage area, to an image data set being created on a disk, one channel-line of the image, where the line specifications have been provided in a NEXTL. Each subsequent IPUT outputs the data for the subsequent channel-line as specified in the NEXTL.

```
IPUT ISCBNAME,USERAREA
```

ISCB - Creates a table of image characteristics and specifications, based on user-supplied parameters, which become the psuedo-label for a new image data set. This information provides the vital statistics needed by the user in interfacing with IDAM and its macros and by the IDAM algorithms in either creating a new image or accessing an old image.

```
ISCBNAME ISCB (For input, no parameters should
be specified.)
```

ITRUNC - Is used in the load application programs to specify the actual number of lines of data which were placed in an image data set so that the unused space allocated by IOPEN can be freed when the ICLOSE is issued.

```
ITRUNC ISCBNAME,GPR4,213 (GPR4 contains the
last line number.)
```

NEXTL - Is used to identify the next (i.e., first) line to be processed by the next IGET or IPUT. Other NEXTL parameters identify which image bands are to be processed, the increment to be used in skipping lines, the number of the first pixel, the number of pixels and the skip factor (if some pixels are to be skipped) of the pixels within each line that are to be processed.

```
NEXTL ISCBNAME,LINE1,BANDLIST,PXCHNGE=(PIX1,
      PIXNUM,SKIP)
```

VII. IDAM IMAGE CREATION EXAMPLE

The following example builds an image on disk via IDAM. The image has four channels, 117 lines, and 196 pixels per line. The lines are output in order.

```
Build ISCB
        IOPEN ISCBNAME,OUTPUT,IMAGNAME
Select lines, channels, and pixels to be out-
put (for this example ALL)
        NEXTL LIST=NLIST
Build line for output in area named DATA.
        IPUT ISCBNAME,DATA
Branch back to build next line
        ICLOSE ISCBNAME
Exit program
ISCBNAME ISCB   MFANC=3060,USERCOM=1800,
                BANDS=(1,2,3,4),LINE1=1
                LINECNT=117,PIXEL1=1,PIXCNT=196
NLIST    DC     A(ISCBNAME)
         DC     X'F000000000000000' BAND1,2,3,4
         DC     H'1'          LINE SKIP FACTOR
         DC     H'1'          FIRST PIXEL
         DC     H'196'        PIXEL COUNT
         DC     H'1'          PIXEL SKIP FACTOR
         DC     H'1'          LINE NUMBER
         DC     X'1'          BYTE PIXELS
IMAGNAME DC     CL12'MYIMAGE'
DATA     DS     196X          DATA AREA
```

VIII. IDAM IMAGE UTILIZATION EXAMPLE

This example reads various segments of the image built above. The NEXTL list shown is an example which could be modified by the program if more than one field were to be processed. The list illustrates the IDAM facility to select a subset of channels, lines, and pixels within a line.

```
        IOPEN ISCBNAME,INPUT,IMAGNAME
Select subset to be processed
        NEXTL LIST=NLIST
        IGET ISCBNAME,DATA
Process one channel-line
Loop back to get next channel-line
Loop back to select next subset
        ICLOSE ISCBNAME
Exit
ISCBNAME ISCB
NLIST   DC    A(ISCBNAME)
        DC    X'4000000000000000' BAND 2 ONLY
        DC    H'1'
        DC    H'23'          FIRST PIXEL
        DC    H'14'          NUMBER OF PIXELS
        DC    H'1'
        DC    H'75'          FIRST LINE
        DC    X'2'           HALF WORD PIXELS
IMAGNAME DC   CL12'MYIMAGE'
DATA    DS    14H
```

IX. EXTENDED CAPABILITIES

The examples and descriptions in this paper do not illustrate all of the capabilities of IDAM. Three of the most commonly used extended capabilities are discussed below.

Pixel core format. IDAM has the capability to format data in the users data area in any of three ways. Each byte of data on the disk may be mapped into one, two, or four bytes in core. When two or four byte pixels are requested, the data byte is right justified with leading zeros in the larger data area. This reformatting is very useful when arithmetic operations are to be performed on the data.

Pixel and line skipping. IDAM has the capability to retrieve only every m-th line and/or n-th pixel from a data set under NEXTL control. Also, when the data is stored, only every m-th line and/or n-th pixel may be output and the pseudo-label will reflect the skip factors. Thus, the original line and pixel numbers may be used to reference the data when it is reloaded. The skipping on retrieval is very useful for display of large images.

Buffer parameters. IDAM, via the IOPEN macro, gives the user the capability to specify the size and number of input buffers to be used. This allows the user to trade off speed versus storage in his application.

X. IDAM EFFICIENCY

By means of the NEXTL macro, the user supplies IDAM with all of the information needed in order to determine which data is to be retrieved first, and the pattern in which subsequent data is to be retrieved. This information is used by IDAM to anticipate I/O demands. As soon as the NEXTL macro is processed, IDAM prepares and executes a channel program to fill a buffer with data from the specified starting point and in the specified pattern. When the first buffer is full, a channel program is prepared and executed to fill the next buffer with data, picking up where the first channel program left off and continuing in the same pattern. Thus, when the application program asks for the next channel-line of data via an IGET macro, the data is already in main core and can be moved at main core data rates into the user's storage using the pointers that were stored in the buffer header when it was filled. As soon as an IGET macro is executed which uses the last channel-line of data in a buffer, IDAM refills it while the other buffer is being processed. This process continues until the end of the data in the image is reached, until another NEXTL macro is issued and a different pattern at a new starting location is begun, or until no more data is processed out of the buffers by the user's program.

In a similar manner, when an image is being created, the data from each IPUT is accumulated in a buffer until it is filled and then, while the other buffer is accumulated, IDAM prepares and executes a channel program to transfer the first buffer of data to the disk.

XI. IDAM PROBLEMS

Over the years that IDAM has been used there has been one problem which has consistently caused trouble. That problem is one of timing in executing a channel program. If it takes the I/O channel longer to resolve the core location for a segment of data from a disk than it takes the disk read-head to scan that segment of data, then the "reading" gets ahead of the "resolving," and a chaining check occurs in the execution of the channel program. The situation occurs when less than 4 bytes (or 8 bytes, depending on the channel) of data is being read or if full (or double) word boundaries are not being maintained as buffer starting points for each read instruction in the channel program. At least one of these situations and sometimes both are seemingly forced to occur when narrow fields, less than 4 (8) pixels wide, are being processed. This situation is aggravated by the fact that track sizes are often not a multiple of 4 (much less 8) bytes, and thus buffer address resolution has to be put back in sync with word (double word) boundaries at the end of each track. Not until recently has logic been developed and verified to solve these problems while maintaining the optionzation outlined in the IDAM TECHNIQUE. So IDAM has, until recently, forced a limitation on ERIPS that fields be a minimum of 4 (8) pixels wide.

A second problem has occurred in a special study in which IDAM was being used to retrieve a large amount of data into some very large user-specified buffers. The result was a channel program which was longer than 2K. The study was being

performed on a virtual memory machine with 2K sized pages. When this channel program was paged out and back in, in the process of core management, Transfer in Channel (TIC) commands were inserted at the page boundary interface. When the channel program was executed, the TIC commands caused execution timing problems similar to those above, again resulting in chaining checks. This problem was solved by forcing the channel program to be formed and executed in a fixed-page region.

XII. CONCLUSION

The capability provided by IDAM and its macros has been used in research activities where flexibility in data selection and ease of utilization for the application programmer was essential. Just as importantly, the ERIPS is also the basic system for a production-oriented project in which large volumes of data are being processed in a timely fashion. IDAM has solved the ERIPS image data storage and retrieval problem so efficiently that, unlike most image processing systems, ERIPS is not I/O bound but rather CPU bound in most applications.